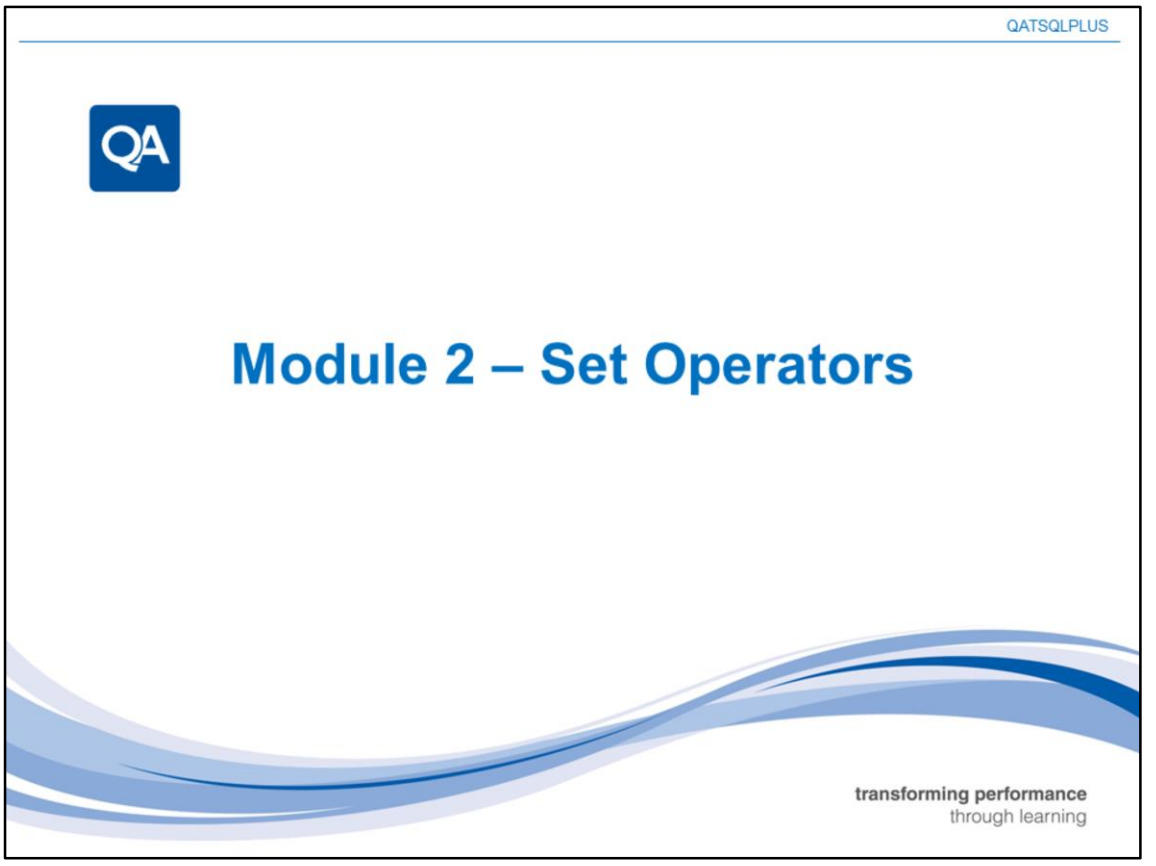
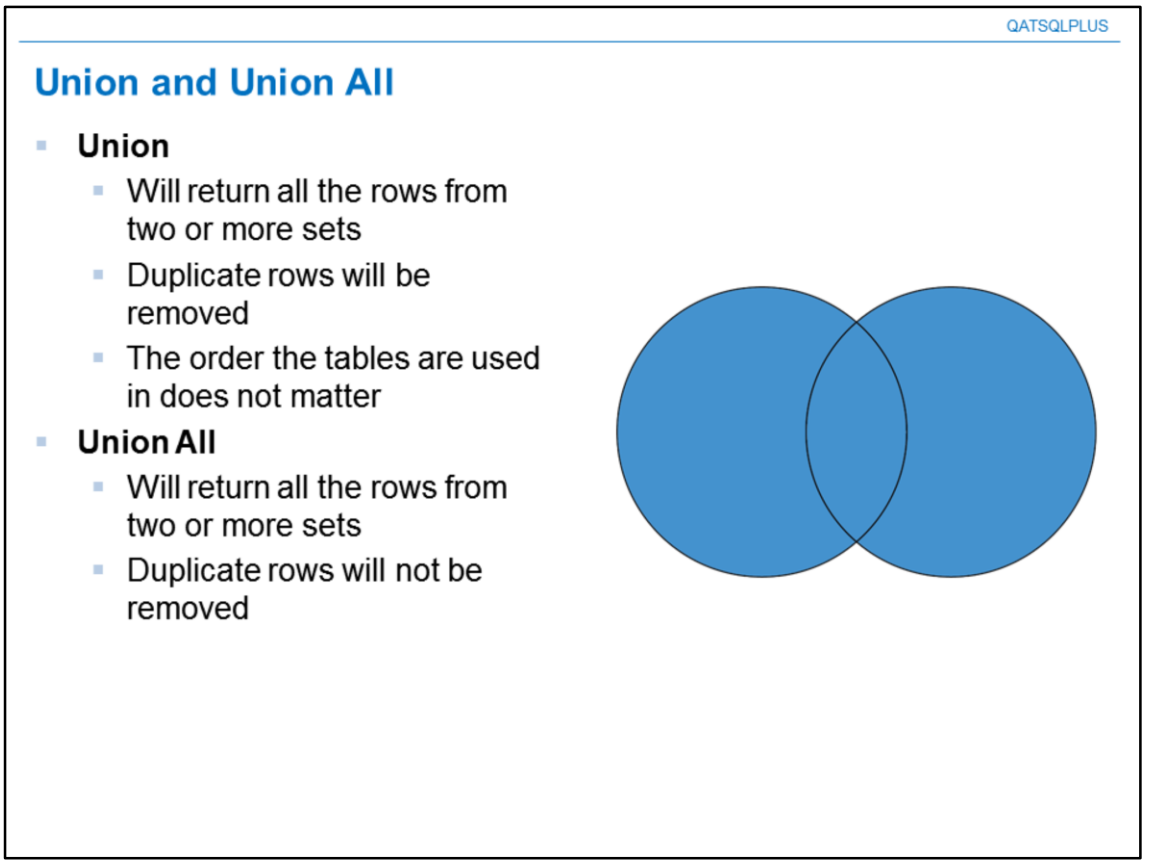

QATSQLPLUS
Module 2 – Set Operators
transforming performance
through learning

Set Operators

- Union
- Intersect
- Except
- Outer and cross apply



Union and Union All

- Union
  - Will return all the rows from two or more sets
  - Duplicate rows will be removed
  - The order the sets are used in does not matter

Example

Dateset 1 : Names = Andrew, Bruce, Charles, David

Dateset 2 : Names = Andrew, Charles, Ellis

Result of Dataset 1 UNION Dataset 2 = Andrew, Bruce, Charles, David, Ellis

- Union All
  - Will return all the rows from two or more sets
  - Duplicate rows will be not removed
  - The order the sets are used in does not matter

Example

Dateset 1 : Names = Andrew, Bruce, Charles, David

Dateset 2 : Names = Andrew, Charles, Ellis

Result of Dataset 1 UNION ALL Dataset 2 = Andrew, Andrew, Bruce, Charles, Charles, David, Ellis

- Applies to both Union and Union All:
  - The sets included must have the same number of columns and each column must have compatible data types
  - The ORDER clause can only be used after the Union / Union All has been performed

Union and Union All

Command outline:

```
SELECT Columns FROM source1
UNION [ALL]
SELECT Columns FROM source2
```

Demonstration:

```
SELECT 'Product' as LineType, Name, ProductSubcategoryID, ProductID,
ListPrice
    FROM Production.Product
    WHERE ProductSubcategoryID IS NOT NULL
UNION
SELECT 'Subcategory',Name,ProductSubcategoryID,NULL,NULL
    FROM Production.ProductSubcategory
ORDER BY ProductSubcategoryID, ProductID
```

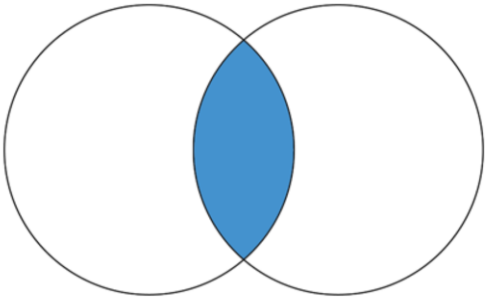
QATSQLPLUS

Intersect

- **Intersect**
  - Returns all rows that exist in both sets
  - The order the tables are used in does not matter

Notes:

- The sets included must have the same number of columns and each column must have compatible data types
- The ORDER clause can only be used after the intersect has been performed



Intersect

- Returns all rows that exist in both sets
- The order the sets are used in does not matter
- The sets included must have the same number of columns and each column must have compatible data types
- The ORDER clause can only be used after the intersect has been performed

Example

Dateset 1 : Names = Andrew, Bruce, Charles, David

Dateset 2 : Names = Andrew, Charles, Ellis

Result of Dataset 1 INTERSECT Dataset 2 = Andrew, Charles

Intersect

Command outline:

```
SELECT Columns FROM source1
INTERSECT
SELECT Columns FROM source2
```

Demonstration:

```
SELECT ProductID, Name, ListPrice
    FROM Production.Product
    WHERE Listprice > 1000
INTERSECT
SELECT ProductID, Name, ListPrice
    FROM Production.Product AS P
    WHERE EXISTS(
        SELECT SUM(OrderQty)
        FROM Sales.SalesOrderDetail AS SOD
        WHERE P.ProductID = SOD.ProductID
        GROUP BY SOD.ProductID
        HAVING SUM(OrderQty) > 1000
    )
```

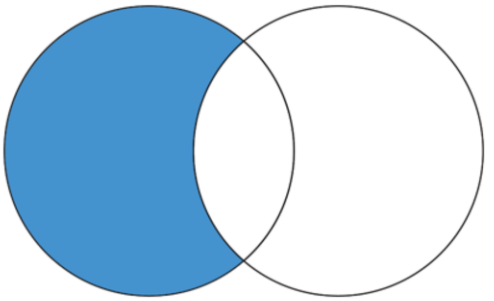
QATSQLPLUS

Except

- **Except**
  - Returns all rows that exist in set1 that do not exist in set2
  - The order the tables are used in does matter

Notes:

- The sets included must have the same number of columns and each column must have compatible data types
- The ORDER clause can only be used after the intersect has been performed



Except

- Returns all rows that exist in the first set, but do not exist in the second set
- The order the sets are used in does matter
- The sets included must have the same number of columns and each column must have compatible data types
- The ORDER clause can only be used after the intersect has been performed

Example

Dateset 1 : Names = Andrew, Bruce, Charles, David

Dateset 2 : Names = Andrew, Charles, Ellis

Result of Dataset 1 EXCEPT Dataset 2 = Bruce, David
Result of Dataset 2 EXCEPT Dataset 1 = Ellis

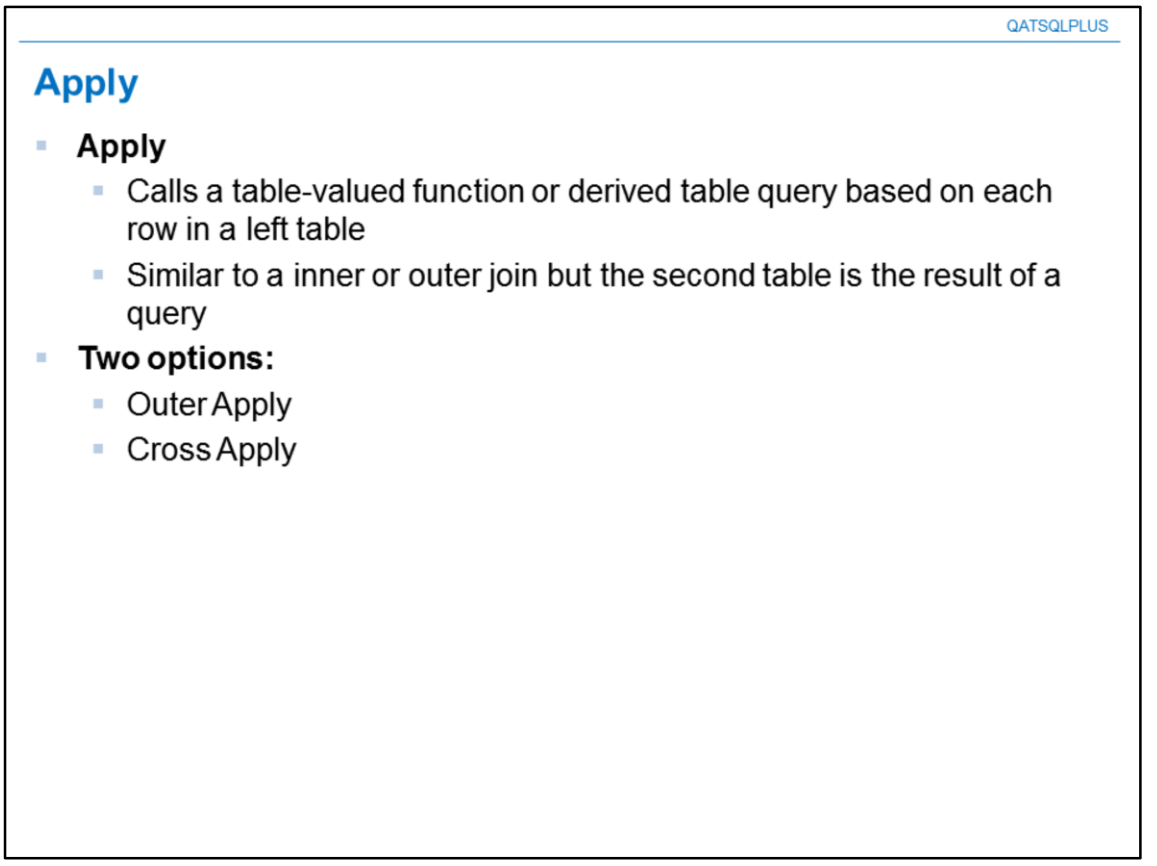
Except

Command outline:

```
SELECT Columns FROM source1
EXCEPT
SELECT Columns FROM source2
```

Demonstration:

```
SELECT ProductID, Name, ListPrice
    FROM Production.Product
    WHERE Listprice > 2000
EXCEPT
SELECT ProductID, Name, ListPrice
    FROM Production.Product AS P
    WHERE EXISTS(
        SELECT SUM(OrderQty)
        FROM Sales.SalesOrderDetail AS SOD
        WHERE P.ProductID = SOD.ProductID
        GROUP BY SOD.ProductID
        HAVING SUM(OrderQty) > 1000
    )
```



Apply

- Calls a table-valued function or derived table query based on each row in a left table or query
- Similar to an inner or outer join, but the second table is the result of a query that takes columns from the first table as parameters, which means that it can be more flexible

There are two forms of APPLY: CROSS APPLY and OUTER APPLY.

- **CROSS APPLY** returns only rows from the outer table that produce a result set from the table-valued function
- **OUTER APPLY** returns both rows that produce a result set, and rows that do not, with NULL values in the columns produced by the table-valued function

Apply

Command outline (Query):

```
SELECT Columns
    FROM InitialTable
        CROSS or OUTER APPLY (SELECT Columns
                    FROM SomeTable
                    WHERE Conditions which can reference InitialTable) AS Alias
```

Command outline (TVF):

```
SELECT Columns
    FROM InitialTable
        CROSS OR OUTER APPLY TVFName(columns as parameters) AS Alias
```

Demonstration:

```
SELECT PS.Name, TS.NumberOfProducts
    FROM Production.ProductSubcategory AS PS
        CROSS APPLY (
            SELECT COUNT(*) AS NumberOfProducts
            FROM Production.Product AS P
            WHERE P.ProductSubcategoryID = PS.ProductSubcategoryID
            GROUP BY P.ProductSubcategoryID
            HAVING COUNT(*) > 10
        ) AS TS
```

QATSQLPLUS

Exercise